Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 19 (2013) 1130 – 1135

Procedia Computer Science

The 3rd International Symposium on Internet of Ubiquitous and Pervasive Things (IUPT)

Device-Based Isolation for Securing Cryptographic Keys ☆

Karim O. Elish, Yipan Deng, Danfeng (Daphne) Yao, Dennis Kafura

Department of Computer Science, Virginia Tech, Blacksburg 24060, USA

Abstract

We describe an effective device-based isolation approach for achieving data security. We show its use in protecting the secrecy of highly sensitive data that is crucial to security operations, such as cryptographic keys used for decrypting ciphertext or signing digital signatures. Private key is usually encrypted in its storage when not used; however, when being used, the plaintext key is loaded into the memory of the host for access. We present a novel and practical solution and its prototype called *DataGuard* to protect the secrecy of the highly sensitive data through the storage isolation and secure tunneling enabled by a mobile handheld device. DataGuard can be deployed for the key protection of individuals or organizations. We implement three prototypes and conduct extensive experiments to evaluate the feasibility and performance of DataGuard. The results show that our approach performs well without significant overhead.

Selection and peer-review under responsibility of Elhadi M. Shakshuki

Keywords:

data security, confidentiality, cryptography, device-based isolation, mobile device

1. Introduction

In the seminal work on computer virus [1], Cohen described isolationism as a potential prevention against the propagation of viruses (on multi-user computers), which refers to no dissemination and no sharing of information across information boundaries. However, absolute isolation clearly would significantly hinder the usefulness of the computing environment. Yet, partial isolation can be strategically implemented to realize specific security goals. For example, Borders *et al.* proposed a solution for achieving data confidentiality that requires disconnection from the network when the data is being accessed [2].

We describe a practical and powerful device-based isolation approach for information security and demonstrate its application in preserving the confidentiality of cryptographic keys. The confidentiality of cryptographic keys is the security foundation of virtually all cryptographic/security protocols, and is essential for individuals and organizations including certificate authorities (e.g., Symantec). Device-based isolation is defined by us as isolating the storage and operations related to data with different security requirements (e.g., confidentiality requirement) through multiple computing devices. In addition, the isolation should not hinder the use and access of the data for practical applications.

☆This work has been supported in part by NSF grant CAREER CNS-0953638 and ONR grant N00014-13-1-0016.

Email address: {kelish, yipanvt, danfeng, kafura}@cs.vt.edu (Karim O. Elish, Yipan Deng, Danfeng (Daphne) Yao, Dennis Kafura)

Selection and peer-review under responsibility of Elhadi M. Shakshuki

doi:10.1016/j.procs.2013.06.160

The novelty of our work is the design and demonstration of how inexpensive and pervasive personal computing devices can be specialized and used for security. We point out that the device isolation and function specialization enable the achievement of high data assurance in applications. These new designs are feasible and practical and benefit from the recent ubiquitous computing reality.

In a strong adversary model, the host and its kernel may be compromised by attackers or malicious software, and thus the confidentiality of the host memory – including cryptographic keys stored there – cannot be guaranteed. For example, simply storing the plaintext private key on a removable medium (eg., a regular USB drive) is not a complete solution because the key still needs to be loaded to the host for use. For this reason, the cryptographic (private) key is usually encrypted on external storage when it is not being used. When needed, the encrypted key is decrypted (which typically requires the user to enter a passphrase). The plaintext key is then loaded into the memory of the host for access.

A number of techniques have been developed to preserve the confidentiality of cryptographic keys. Using a tamper-resistant smart card for the key storage requires a special reader, which may not be available. In a more complex solution, the host may be equipped with an on-chip trusted platform module (TPM) that allows the host to attest its system integrity to a trusted server when the plaintext key is loaded; the encrypted private key can only be decrypted when the host's system integrity is verified. However, one major limitation of this approach is that TPM cannot prevent any run-time compromise taking place after the attestation [3]. Thus, the keys may be exposed. In general, the fundamental and practical security problem of protecting the secrecy of the private key is not adequately addressed in the security literature.

In this paper, we present a novel and practical technique to enhance the confidentiality of private keys. Our design separates private keys from an untrusted host to a special-purpose, trusted handheld device, while still enabling the host to use the key (e.g., for decryption and signing). Our approach recognizes that ensuring system integrity – the assurance that applications and the host's operating system including the kernel have not been compromised by attackers or malicious software – of a general-purpose computer with Internet connectivity is difficult in general. However, a dedicated special-purpose computing device is relatively easier to secure and less susceptible to attacks. Such a device may be an inexpensive smartphone or tablet computer. We use a trusted device for dedicated key storage and cryptographic operations, which allows the key to be used without being exposed to the untrusted host. Accordingly, our solution can be easily deployed by any entities (individuals or organizations) using the public key cryptosystems to enhance their protection of private keys.

Our device-based isolation framework – named DataGuard – provides two security functions: *i)* it protects highly sensitive cryptographic keys by isolating them from the untrusted general computing environment, and *ii)* it supports the secure use of the key through a protocol by which the host may request a message to be signed with the protected private key. Without our device-based isolation of secret keys, the cleartext key value would exist in the memory of the host when it is being used, and thus a compromised host may lead to key compromise. In our prototypes, we show several independent instantiations of DataGuard with various types of machines, including a smartphone, a tablet computer, and a Linux box for embedded system. More details can be found in our technical report [4].

2. Design Overview

2.1. DataGuard Architecture

Our device-based isolation design consists of two main components: *i)* a host running a DataGuard daemon, and *ii)* an external trusted device running a DataGuard application, which are described as follows.

1. **Host** is a personal computer which is vulnerable to malware attacks. However, the host's owner needs to utilize the private cryptographic keys to decrypt ciphertext or sign messages. Hence, we need to provide the security enhancement to this host by protecting the secret keys from being compromised. **DataGuard daemon** is a software application used to connect the host with the external trusted device. It initiates a communication channel with the trusted device and uses this channel to execute the commands requested from the DataGuard application running on the trusted device.
2. **External trusted device** is an external device used to enhance the data security of the host by storing the cryptographic secret keys of the host and performing the cryptographic operations. DataGuard

is deployed on this device. The selection of the trusted device should support the following criteria in order to be used in our framework: *i)* it should have the capability to perform computation, *ii)* it should have storage space, *iii)* it should have the ability to communicate with other devices, and *v)* it should have I/O capability with a display screen.
Any device that possesses these characteristics can be used as a trusted device in our framework.
DataGuard application is a software application running on the external trusted device. It is used to communicate with the host and performs the security-related cryptographic operations (namely decryption and signing) which are described in details in Section 4.

2.2. Security Goals, Threat Model, and Assumptions

- **Security goals:** The primary goal of our framework is to enhance the data confidentiality of a host by protecting its sensitive data from being compromised by malicious attacks. In particular, our security goals are three-fold: *i)* to ensure the confidentiality of cryptographic private keys, *ii)* to ensure the confidentiality of ciphertext being decrypted, and *iii)* to ensure the integrity of digital signature produced.

- **Threat model and security assumptions:** In our threat model, we assume that the host including the user-space and kernel-space data and code are not secure and vulnerable to malicious attacks. Specifically, the attacker may attempt to: *i)* gain knowledge of a private key, *ii)* gain knowledge of the message encrypted with the corresponding public key, and *iii)* forge digital signatures on behalf of the legitimate key owner.

  We assume that the external device's data and code are trusted and not compromised. This assumption is reasonable, as the device can be a special-purpose computing device with no or limited network connectivity and limited permitted operations.

A general purpose computer may be vulnerable to malicious attacks as it is typically connected to the Internet. DataGuard provides a secure communication channel between the external device and the host. In this case, the external device is more isolated and easier to secure compared to the conventional approach. The external device is not exposed directly to the Internet. We refer to the external device as a *trusted device*. The external device is easier to secure compared to a general purpose computer because of two reasons as follows: *i)* it only securely communicates to the host, and *ii)* it only provides one specific key usage service, its data and code are small as a result of limited operations.

Therefore, our assumption on the integrity of the trusted device is practical. The storage and usage environments of the cryptographic keys that we aim to protect are isolated from the general computing environment. This property is achieved because the keys are stored and operated on the trusted device, and never appear on general purpose computers (i.e., the host).

3. Protocols for Accessing Private Keys

In a device-based isolation architecture, cryptographic private keys are stored on a trusted dedicated device, as opposed to a general-purpose networked computer (i.e., host), for the confidentiality of the key. In order to use the key for decryption or signing, the host needs a protocol to utilize the private key *without* learning the key value. In this section, we describe two protocols for using private keys in our DataGuard framework: *i)* decrypting the ciphertext received by the host, and *ii)* signing an outgoing message.

Secure decryption protocol. This protocol allows the DataGuard user (Alice) to decrypt the ciphertext received from the Internet. The sender of the message (e.g., Bob) encrypts a secret message using Alice's public key and then sends it to Alice. Alice's host receives the encrypted message and sends it to the trusted device for decryption using the private key stored in the trusted device. Then, the decrypted/original message is displayed on the screen of the trusted device to ensure the confidentiality of the original message. Figure 1 depicts the secure decryption protocol in DataGuard. The detailed operations are as follows:

1. Alice's host receives the ciphertext C from Bob who encrypts the message M with the public key of Alice to ensure the confidentiality of M.

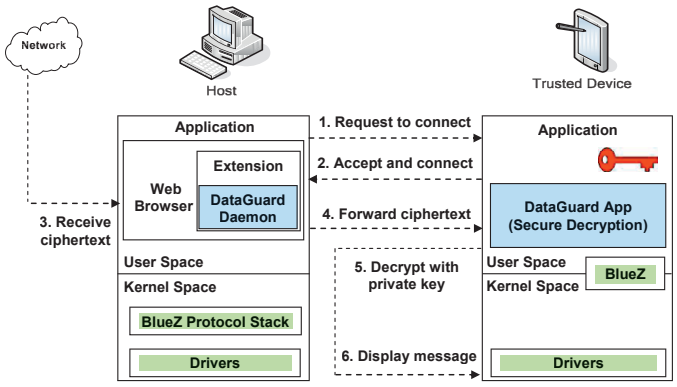

Fig. 1. Secure decryption procedure in DataGuard. Red key is the private key of the DataGuard user.

2. The DataGuard daemon on Alice's host initiates a request for connecting to the trusted device.
3. The trusted device provides a PIN/passkey to verify its identity before connecting to Alice's host. Upon the verification, the trusted device is paired with Alice's host and the connection is established.
4. Alice's host forwards ciphertext C to the secure decryption application running on the trusted device.
5. The secure decryption application receives C, and decrypts C with Alice's private key K stored in the trusted device to obtain the message M. Alice may need to enter her passphrase on the trusted device in order to decrypt the private key.
6. Message M is displayed on the screen of the trusted device.

One application of this secure decryption protocol mechanism is secure email, i.e., decrypting encrypted email messages. To protect the email confidentiality usually the sender and receiver use end-to-end email encryption tool like openPGP. Our solution can also be used to protect email confidentiality. For example, the DataGuard user receives an encrypted email. The host relays the encrypted email to the trusted device. The trusted device performs the decryption and displays the plaintext message on its screen. In addition, the sender can use our approach to encrypt the email message before sending it. Thus, our solution can protect email confidentiality even if the attacker controls the host/computer or has access to the mail server.

Secure signing protocol. In this protocol, Alice uses her external trusted device to digitally sign a message. Whenever Alice needs to send a signed message to Bob, the message is forwarded to the trusted device for signing before Alice sends the message to Bob. The detailed operations of secure signing protocol can be found in [4]. One application of the secure signing protocol is electronic purchasing where a customer sends a signed purchase order to a vendor.

4. Implementation and Evaluation

DataGuard architecture requires the establishment of a secure channel between the host and the trusted device. Therefore, we implemented and supported two types of communication methods: *i)* wired using universal serial bus (USB) and *ii)* wireless using Bluetooth. For Bluetooth communication in our prototype, we chose BlueZ which is the Linux Bluetooth Protocol Stack.

We developed two Android-based applications to support the security protocols presented in this paper, namely *i)* a secure decryption application which performs the secure decryption service and *ii)* a secure signing application which performs the digital signing service. We support different algorithms (RSA and AES) with different key lengths for decryption, and DSA and RSA with different key lengths for signing. We developed two Chrome extensions, namely a secure decryption extension and a secure digital signing extension, to demonstrate the practicality of our approach. Due to the space limitation, the implementation details of DataGuard framework can be found in [4].

We conducted several experiments to evaluate the efficiency of our DataGuard prototypes, including the running time of cryptographic operations on the trusted device. We used a Lenovo T410 laptop as a host

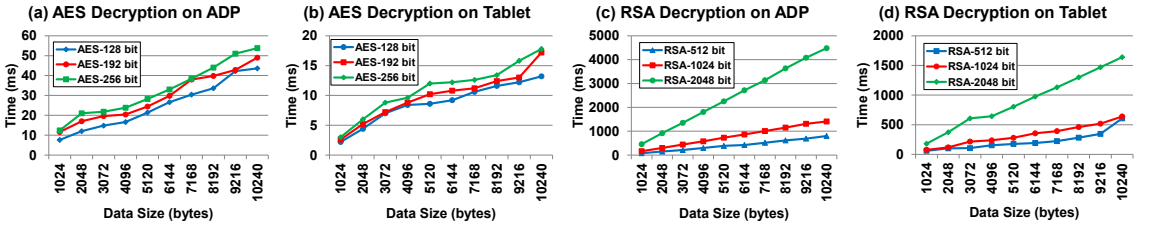

Fig. 2. Decryption performance of AES and RSA on handheld devices.

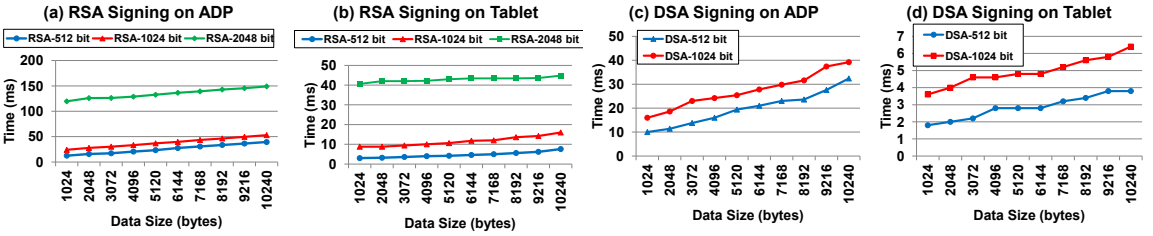

Fig. 3. Digital signing performance of RSA and DSA on handheld devices.

which has 2.40GHz dual core Intel Core i5 processors, 2GB memory, a Bluetooth chipset compatible with Bluetooth 2.1 specification, and Ubuntu 10.04 LTS. We realized our prototype and evaluations on three different trusted handheld devices: a Linux box device (ADVANTECH Linux Box-PC ARK-1360) designed for embedded systems, Android Development Phone (ADP), and Samsung Galaxy Tab 10.1. Android Development Phone (ADP) is a smartphone that runs Google Android OS 2.1 and has a 576MHz processor with 512MB ROM, 288MB RAM, and Bluetooth 2.0. Samsung Galaxy Tab 10.1 is a tablet computer that runs Google Android OS 3.1 and has a dual-core 1GHz processor, 32GB ROM, 1GB RAM, and Bluetooth 2.1. All of them meet the requirements for the trusted device described in Section 2.1.

For completeness, we implemented and evaluated all relevant operations of both public-key and symmetric-key schemes in DataGuard, including encryption/decryption and signing/verification. Figure 2 shows the decryption performance of different ciphertext sizes on two handheld devices by using AES (128-bit, 192-bit and 256-bit keys) and RSA (512-bit, 1024-bit, and 2048-bit keys), respectively. Figure 3 shows the signing performance on two handheld devices by using DSA (512-bit, 1024-bit) keys and RSA (512-bit, 1024-bit, 2048-bit) keys, respectively. Our experimental results show feasible performance of DataGuard as all cryptographic operations can be performed efficiently on the external trusted handheld devices without significant overhead, and the communication bandwidth of Bluetooth is sufficient for transmitting large amounts of data (e.g., ciphertext or digital signature) in a timely fashion. More evaluation results can be found in [4].

Security analysis. *Key confidentiality.* Our framework provides secure storage for cryptographic private keys by isolating and storing them on a dedicated trusted device instead of keeping them on the untrusted host. All cryptographic operations involving the private keys take place in the external trusted device as well. Therefore, the values of private keys are not revealed to the host and the confidentiality of private keys is preserved. *Message confidentiality.* The confidentiality of the message is preserved because the decryption is performed and displayed on the trusted device. The host only has the access to the ciphertext. *Message integrity.* The integrity of the message can be preserved and signature cannot be forged, as the digital signature can only be generated by the trusted device uniquely possessing the private key. Due to the space limitation, the detailed discussion on security analysis of DataGuard can be found in [4].

5. Related Work

McCune et al. [5] proposed a framework called Bump in the Ether (BitE) to ensure the integrity of sensitive user input to the applications running on the host. In [6], the authors introduced a Bumpy system

that protects the sensitive user input for web applications by processing the input in an isolated code module on the user's system, where they can be processed for a remote webserver. BitE and Bumpy systems address the integrity of user input to the applications, while our work addresses the confidentiality requirement of secret data (cryptographic keys) used in sensitive operations (decryption and signing). Storage Capsules system is a recent approach introduced by Borders et al. [2] to protect confidential files on a personal computer. It allows a compromised host to securely view and edit sensitive files by isolating the primary operating system in a virtual machine and disabling network and other device output before it is accessing confidential files. Our work uses a separate device for isolation to achieve the data confidentiality, while Storage Capsule performs the isolation on the same machine. Zic and Nepal [7] designed and implemented a prototype personal trusted device called the Trust Extension Device (TED) that provides users with a portable trustworthy environment for performing transactions on computer which is connected to Internet. Also, Nepal et al. [8] designed and built a mobile and portable Trusted Computing Platform (TCP) based on Trusted Computing Group (TCG) specification. Their solution provides a TCP on a USB device to enable the mobility and portability of trust for enterprise applications. The solutions in [7] and [8] do not provide a complete isolation from untrusted host as their proposed device does not have any input/output devices on its own, instead it relies on the input/output devices of the untrusted host machine. Some other solutions have used mobile handheld devices to provide secure authentication techniques [9, 10], provide secure remote access to the user's home computing environment [11], propose a realistic mobile ticket system [12], and enhance web browsing security on public terminals [13].

6. Conclusions and Future Work

With the proliferation of personal handheld computing devices, such as smartphones and tablet computers, we proposed a device-based isolation approach and its prototype called *DataGuard* to protect the secrecy of the highly sensitive data (namely cryptographic keys) through the storage isolation and secure tunneling. Our solution does not require highly specialized (system-specific) services and hence it would be easily ported to other platforms. We conducted extensive experiments to evaluate the feasibility and performance of DataGuard. The results show that our approach performs well without significant overhead. For future work, we plan to perform more systematic studies to evaluate the usability of our approach to find out users' experiences in using an external device to store secret keys and perform cryptographic operations.

References

[1] F. Cohen, Computer viruses theory and experiments, Computers and Security 6 (1987) 22 – 35.
[2] K. Borders, E. V. Weele, B. Lau, A. Prakash, Protecting confidential data on personal computers with storage capsules, in: 18th USENIX Security Symposium, 2009, pp. 367–382.
[3] S. Garriss, R. Cáceres, S. Berger, R. Sailer, L. van Doorn, X. Zhang, Trustworthy and personalized computing on public kiosks, in: Proc. of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys), 2008, pp. 199–210.
[4] K. O. Elish, Y. Deng, D. Yao, D. Kafura, Device-based isolation for securing cryptographic keys, Technical Report TR-12-21, Virginia Tech (2012).
[5] J. M. McCune, A. Perrig, M. K. Reiter, Bump in the ether: A framework for securing sensitive user input, in: USENIX Annual Technical Conference, General Track, 2006, pp. 185–198.
[6] J. M. McCune, A. Perrig, M. K. Reiter, Safe passage for passwords and other sensitive data, in: Proc. of the Network and Distributed System Security Symposium (NDSS), 2009.
[7] J. Zic, S. Nepal, Implementing a portable trusted environment, in: Future of Trust in Computing, 2009, pp. 17–29.
[8] S. Nepal, J. Zic, D. Liu, J. Jang, A mobile and portable trusted computing platform, EURASIP Journal on Wireless Communications and Networking 75 (2011) 1–19.
[9] M. Mannan, P. C. van Oorschot, Using a personal device to strengthen password authentication from an untrusted computer, in: 11th International Conference on Financial Cryptography and Data Security, and 1st International Workshop on Usable Security, 2007, pp. 88–103.
[10] G. Starnberger, L. Froihofer, K. M. Göschka, QRTAN: Secure mobile transaction authentication, in: Proc. of the 4th International Conference on Availability, Reliability and Security (ARES), 2009, pp. 578–583.
[11] A. Oprea, D. Balfanz, G. Durfee, D. K. Smetters, Securing a remote terminal application with a mobile trusted device, in: 20th Annual Computer Security Applications Conference (ACSAC), 2004, pp. 438–447.
[12] Y.-Y. Chen, C.-L. Chen, J.-K. Jan, A mobile ticket system based on personal trusted device, Wireless Personal Communications: An International Journal 40 (2007) 569–578.
[13] R. Sharp, A. Madhavapeddy, R. Want, T. Pering, Enhancing web browsing security on public terminals using mobile composition, in: Proc. of the 6th International Conference on Mobile Systems, Applications, and Services (MobiSys), 2008, pp. 94–105.